CS342 Machine Learning Assignment 2

Iain Souttar, 1505756

March 14, 2018

Abstract

For this project, we were asked to implement different models to segment and identify nuclei in images. The images given varied in structure and composition, so it was vital to keep the model sufficiently general to prevent overfitting. The first step was to analyse the data to identify possible paths for feature engineering and processing techniques. I was able to identify two main types of image which had different colours of cells and general brightness. This allowed me to better identify the background in these images- useful when producing masks through thresholding. An MLP was used both with raw pixel values and engineered features. This was slightly unfit for purpose as explained. A CNN was then created to more accurately predict the cells- and was the best performer.

1 Data Exploration, Feature Engineering, and Segmentation

1.1 Task 1

The task at hand is to produce a model which takes in images and produces masks which convey the position of the cells contained. I began by looking at a few of the images. It was quickly clear that there were at least two types of image; most very dark but others much lighter. The qualitative distinction can be seen in Figure 1.

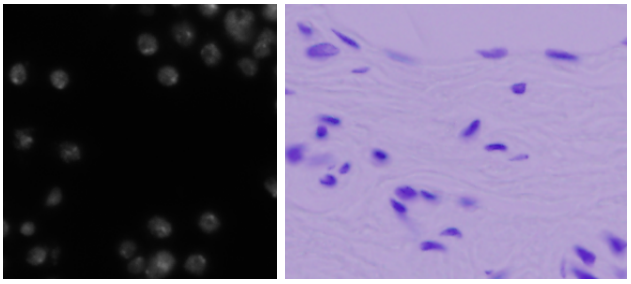

Figure 1: Difference in colour of the Dark (*left*) and Light (*right*) images.

This could be down to difference in method of capturing, magnitude or even type of cell. As a result I looked to plot the red, green and blue values of each image to see if they could be separated this way. Two very distinct classes of image were present so I decided to- before moving on- divide them up. This is

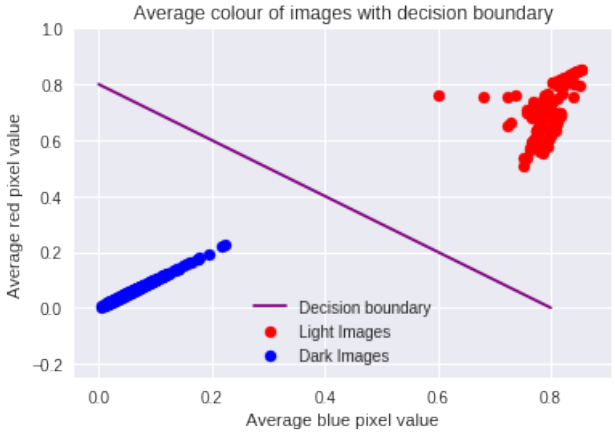

Figure 2: Average red and blue pixel value of images, and the separation.

quantitatively seen in Figure 2. To do this, I created a decision boundary on the red on blue pixel value graph of $red + blue = 0.8$. I made this decision because it seemed to me that the composition of these two types of images were very contrasted and as a result it could be useful further on with post or pre processing.

I wanted to explore this difference a bit more; it seemed significant. This led me to plot the number of nuclei in the corresponding class of image, along with the average grayscale of these cells (Figure 3), to see if there was any difference in these attributes.

You can see from the plots in Figure 3 that the number of nuclei is similarly distributed in the light and dark images. Although there is a higher variance in the light images number of nuclei, it was not worth pursuing since there were less of these so a higher variance comes naturally. However, there is a vast, possibly expected, difference in grayscale of the cells in the images. This plot tells us that the cells in the 'light' images are darker than the background but the opposite is true in the 'dark' images. This proved useful in Task 3, where it was necessary to determine what was background.

1.2 Task 2

The first feature expansion technique I used was data augmentation. I rotated the images by 90 degrees three times each, to quadruple the amount of training data for my MLP. This was especially useful for predicting the lighter images, as there were only 124 of these to train on originally. However, there was a limit to the improvement that it made- it sometimes even worsened the performance of the model. This is perhaps

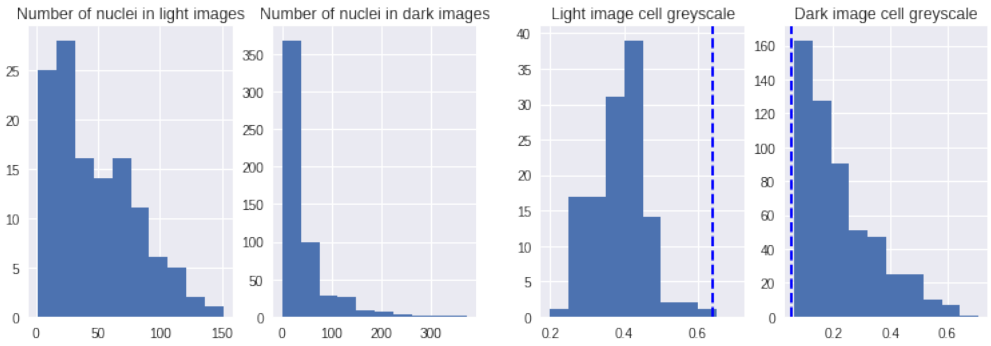

Figure 3: Number of nuclei in each class of image (*left*). Histogram for the greyscale of cells in respective class of image. Dotted line indicates average greyscale of all pixels in that class of images (*right*).

because with rotation and reflection you are not producing more types of images for the model to train on, only more of the same types.

I also used discrete Fourier transforms as a feature engineering technique. This involved applying a high pass filter to the image in the frequency domain. It reduced noise and highlighted the edges. This is useful, particularly in our case of identifying nuclei, since it can display edges of objects well. Decreasing the filter strength allowed lower frequency nuclei to be picked up, but meant that some noise was too. This is an example of having to find the right balance for our particular case. This can be seen in Figure 4.

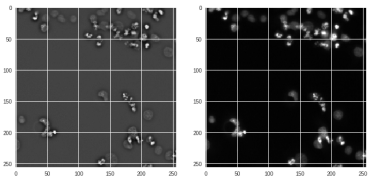

Figure 4: Discrete fourier transforms of two images, one of which misses some cells (*left*) and one that picks them up (*right*).

The third I used was Histogram of oriented gradients. Histogram of oriented gradients computes the change of colour across the pixels and the direction of this change. In terms of greyscale, this means it calculates where the image changes in brightness. This is useful for predictions because it picks up changes in colour between the background and cells. In sci-kit, the HoG function allows you to vary the number of orientations that it assesses. I found that a higher number is better for implementation with the model. This is because, when thinking about the roughly circular shape of cells, we want it to pick up changes in colour from lots of different angles. For images with sharper and straighter lines, a lower number would be sufficient. Of course, too high a number of orientations risks the model picking up lots of noise and overfitting.

1.3 Task 3

I decided to implement a threshold technique, as this is what I had done the most analysis on in the first task. For this I used the guidance of a kernel [1]. The method was to, for each test image, find a 'best' threshold and decide whether each pixel was a cell or not this way. The threshold found was using Otsu's method, which minimises the variance between classes. This includes an assumption that the distribution of greyscale in the images is somewhat bimodal- the cells and the background. It became apparent that there would be a problem in deciding which side of the threshold was background and which was the cells.

Here my work in Task 1 came in handy- I had already split the data into two classes, and seen there was a significant difference in cell colour of each compared to the mean. In the light images, the cells were darker than the average but in the dark images they were lighter. This meant I was able to swap the side of the threshold for which it was decided that pixel is a cell depending on the image being light or dark. This gave me a score of 0.258. Given more time, I would perhaps have experimented with different types of threshold as many of the images have very few cells or very little background- meaning the distribution would be hardly bimodal.

[1] https://www.kaggle.com/stkbailey/teaching-notebook-for-total-imaging-newbies

2 Machine Learning Models: Artificial Neural Networks and Deep Learning

2.1 Task 4

The first thing to note for producing models on this dataset was that there were different sizes of images. Some were more than 4 times the amount of pixels than others. As a result the first step was to resize (after storing the original size) to 256x256. I chose this size because it was both the modal size and the minimum. This minimised the distortion introduced into the images when resizing. They could then all be passed into the same model for training, produce a mask and then be resized to their original dimensions for encoding which was done using methods from a kernel on Kaggle [2]. I greyscaled all images beforehand.

This task was to implement an MLP. Through some experimentation, I ended up using 7 layers, including the input and output layers. I also used a dropout meaning that, randomly, a proportion of the weights calculated in the previous layer are set to 0. This helps to prevent overfitting and limits the amount of parameters of the model. During the experimental phase, I varied the dropout, and I found that increasing the dropout made the model better but only to a certain point, which agrees with the intuition. The last activation layer being 'sigmoid' is due to the attributes of this function. It separates the data and squashes it between 0 and 1, producing probability-like values for each pixel. This then allows us to use a threshold of 0.5 to decide whether the prediction is a cell or background and produce masks.

The MLP was relatively poor compared to just thresholding (in task 3) and the CNN implemented later on. This is because of the structure of an MLP. It produces weights for all pixels in a flattened structure- meaning it is not spatially variant. As a result, the weights of pixels very far away from one influence its predictions. As we can see in the images, this should not be the case because cells seem to have no real relation to each other spatially. I did produce separate models for the light and dark images, but the lack of light images made that model very poor and more than counteracted the positives. In the end I just produced one model because of the relative lack of data.

2.2 Task 5

The three feature engineering/segmentation techniques I implemented were data augmentation, Histogram of oriented gradients and Fourier transforms as spoken about before. In terms of post-processing, I implemented a basic check on the cells predicted to get rid of very small ones (less than 20 in area). This reduced the noise in masks and produced cleaner, more accurate predictions. It improved the score by up to 0.05 in some cases.

I used discrete fourier transforms, as explain in task 2, to highlight edges and identify the shape of nuclei. This produced a score of 0.15 with the MLP, showing some promise as a technique. However, it fared badly with images that had lots of noise as it would pick them up. With more time, I would look to introduce post-processing techniques in order to reduce the affect this has on our final predictions- possibly with filling in cells or getting rid of very small predicted cells.

For data augmentation, I simply appended rotations of all training data, along with the rotated masks. I rotated each image 3 times by 90 degrees, to ensure the borders and shape of the array were preserved. This quadrupled the amount of data for the respective model to train on- allowing higher accuracy and less overfitting because of more data to train on. This, in particular, worked well for the MLP with thresholded inputs- gaining a score of 0.235. However, this was still lower than just thresholding. As mentioned before, this is due to the fact that the MLP is not very well suited to problems where the position of something is important because all pixels contribute to the prediction of each pixel.

The last I used was HoG. As explained more in Task 2, this was implemented to detect changes in colour or brightness in the image- intuitively relevant to detecting nuclei. When used with the MLP, HoG didn't perform very well. This may be because of the limitations of the MLP more than HoG, as it doesn't take into account locality of surrounding pixels. As seen in Figure 6, there was lots of variation when changing the parameters. The more noise in the image, the more it picked up gradients which weren't useful to identifying nuclei and as such predicted some which were just variations in colour of background.

I also used cross validation with this model. This meant that the loss would be minimised over 90 percent of the data, and validated on the remaining 10 percent. This is useful because it can give an idea of how well the model generalises to new data- something which it clearly needs to be able to do when predicting the test images. It helps to identify overfitting as the accuracy would be high but the validation accuracy would be low. K-fold could be introduced but would take a lot longer with complicated models so I opted for just having a 10 percent cross validation.

[2] https://www.kaggle.com/kmader/nuclei-overview-to-submission

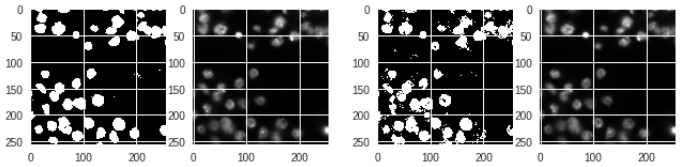

Figure 5: Two HoG predictions of an image, one of which is quite accurate (*left*) and one that picks up lots of noise (*right*, with far higher number of orientations).

2.3 Task 6

The CNN works much better than the MLP for this project. This is due to the ability of the kernel to move across the image, assigning weights to pixel blocks close to the pixel that is being predicted. This means that although a CNN can have extremely large amounts of parameters- they are more likely to be useful and helpful to the model than with an MLP in our case. Again, a dropout is key here, to limit the amount of parameters and prevent overfitting. It can, however, go too high and underfit. This means too many parameters are being discarded and so patterns in the data cannot be recognised.

It produces many amounts of parameters through its convolutional layers, and then reduces these down, making it able to be more general than the MLP while making use of the important parameters. As a result, the model can perform better on new data. In contrast the MLP was not able to do this adequately, hence the lower score.

There are many different options when implementing a CNN. Epochs and cross validation are useful to produce the best model possible. When using a large amount of epochs, it is sometimes useful to have an EarlyStopper which Keras allows you to introduce. This means that if the model does not improve within the specified amount of epochs, the model will stop training and take the best of the previous to predict with. Using a metric similar to the one being assessed allows us to clearly identify and rank certain models. However, there is the risk of overfitting to the metric- meaning that the model produces good results for the metric but undesired predictions overall. Again, cross validation, dropout and other techniques can help to prevent this and ensure the model generalises to new data well.

I also introduced a max pooling layer. This takes parameters from the previous layer and combines them into a single parameter by taking the maximum for outputting. As a result, less parameters are used in the model- reducing the chance of overfitting the data. It also ensures these are the most important- as they have the largest weight. As explained, this helps to keep the model general and able to adapt well to new test data.

3 Progression Graph and Discussion

3.1 Task 7

My progression graph conveys the fact that the thresholding was a reasonable technique. It outperformed all MLP prediction scores. The CNN was the best- but not by a massive amount. The results indicate, as explored in this report, that the MLP was generally unfit for the task required, possibly due to the lack of ability to differentiate between cells near and far, as discussed. The post-processing I had time to implement (a simple check of size of cell) did improve the scores. This encourages me to conclude that post-processing could have a large affect if introduced properly. This would be the avenue I would go down in terms of improving my models immediately- maybe by having a fill in method or better separation between cells. I would also look to begin improving my CNN. It is currently a fairly simple one using just raw pixel values. My best submission on Kaggle was, however, using data augmentation through rotation but only two rotations since the computers could not run it with any more. I would use my other feature engineering techniques to provide the CNN with more data on the images and produce a better model. More data augmentation techniques would also be useful and may even allow the implementation of two separate models for Light and Dark images by providing the necessary data.

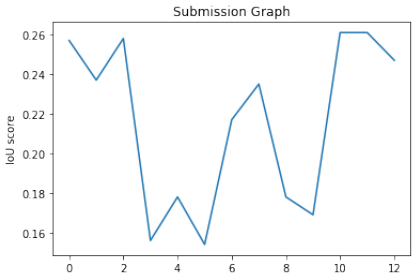

Figure 6: Submission graph. Threshold → Threshold → Threshold with swap → MLP → MLP added layers → MLP too many layers → MLP with data augmentation → MLP data augmentation with threshold → MLP Hog → MLP Fourier → CNN → CNN lower dropout → CNN much lower dropout.